**字符串：**

[1．KMP： 2](#_Toc491270735)

[2.EKMP： 2](#_Toc491270736)

[3．Manacher 4](#_Toc491270737)

[4．最小表示法 5](#_Toc491270738)

[5.Trie 5](#_Toc491270739)

[6.AC自动机 6](#_Toc491270740)

[7.后缀自动机 8](#_Toc491270741)

#### 1．KMP：

void kmp\_pre(char x[],int m,int nexts[]){

int i,j;

j=nexts[0]=-1;

i=0;

while(i<m){

while(-1!=j&&x[i]!=x[j])

j=nexts[j];

nexts[++i]=++j;

}

}

//返回x在y中出现的次数，可以重叠

int nexts[10010];

int KMP(char x[],int m,char y[],int n){//x是模式串，y是主串

int i,j;

int ans=0;

kmp\_pre(x,m,nexts);

i=j=0;

while(i<n){

while(-1!=j && y[i]!=x[j])

j=nexts[j];

i++;j++;

if(j>=m){

ans++;

j=nexts[j];

}

}

return ans;

}

#### 2.EKMP：

//扩展KMP算法

//next[i]:x[i...m-1]与x[0...m-1]的最长公共前缀

//extend[i]:y[i...n-1]与x[0...m-1]的最长公共前缀

void pre\_EKMP(char x[],int m,int next[]){

next[0]=m;

int j=0;

while(j+1<m && x[j]==x[j+1])

j++;

next[1]=j;

int k=1;

for(int i=2;i<m;i++){

int p=next[k]+k-1;

int L=next[i-k];

if(i+L<p+1)

next[i]=L;

else{

j=max(0,p-i+1);

while(i+j<m&&x[i+j]==x[j])

j++;

next[i]=j;

k=i;

}

}

}

void EKMP(char x[],int m,char y[],int n,int next[],int extend[]){

pre\_EKMP(x,m,next);

int j=0;

while(j<n && j<m && x[j]==y[j])

j++;

extend[0]=j;

int k=0;

for(int i=1;i<n;i++){

int p=extend[k]+k-1;

int L=next[i-k];

if(i+L<p+1)

extend[i]=L;

else{

j=max(0,p-i+1);

while(i+j<n&&j<m&&y[i+j]==x[j])

j++;

extend[i]=j;

k=i;

}

}

}

#### 3．Manacher

/\*

\* abaaba

\* i: 0 1 2 3 4 5 6 7 8 9 10 11 12 13

\* Ma[i]: $ # a # b # a # a $ b # a #

\* Mp[i]: 1 1 2 1 4 1 2 7 2 1 4 1 2 1

\*/

char Ma[maxn\*2];

int Mp[maxn\*2];

void Manacher(char s[],int len){

int l=0;

Ma[l++]='$';

Ma[l++]='#';

for(int i=0; i<len; i++){

Ma[l++]=s[i];

Ma[l++]='#';

}

Ma[l]=0;

int mx=0,id=0;

for(int i=0; i<l; i++){

Mp[i]=mx>i?min(Mp[2\*id-i],mx-i):1;

while(Ma[i+Mp[i]]==Ma[i-Mp[i]])

Mp[i]++;

if(i+Mp[i]>mx){

mx=i+Mp[i];

id=i;

}

}

}

char s[maxn];

int main(){

while(scanf("%s",s)==1){

int len=strlen(s);

Manacher(s,len);

int ans=0;

for(int i=0; i<2\*len+2; i++)

ans=max(ans,Mp[i]-1);

printf("%d\n",ans);

}

return 0;

}

#### 4．最小表示法

int GetMin(int s[],int len){//数组开两倍

int i=0,j=1,k=0;

while(i<len&&j<len){

k=0;

while(s[i+k]==s[j+k]&&k<len)

k++;

if(k==len)

return min(i,j);

//改成 < 就是最大表示法啦

if(s[i+k]>s[j+k])//即s1[i->i+k]不会是该循环字符串的“最小表示”的前缀

if(i+k+1>j)

i=i+k+1;

else

i=j+1;

else if(j+k+1>i)

j=j+k+1;

else

j=i+1;

}

if(i<len)

return i;

return j;

}

#### 5.Trie

const int MAX = 26;

const int maxnode = 2e6+7;

const int sigma\_size = 2;

struct Trie{

int ch[maxnode][sigma\_size];

int val[maxnode];

int sz;

void clear(){

sz=1;

memset(ch[0],0,sizeof(ch[0]));

}

int idx(char c){ return c-'a'; }

void insert(char \*s){

int u=0,n=strlen(s);

for(int i=0; i<n; i++){

int id=idx(s[i]);

if(ch[u][id]==0){

ch[u][id]=sz;

memset(ch[sz],0,sizeof ch[sz]);

val[sz++]=0;

}

u=ch[u][id];

val[u]++;

}

}

int search(char \*s){

int n=strlen(s),u=0;

for(int i=0; i<n; i++){

int id=idx(s[i]);

if(ch[u][id]==0)

return 0;

u=ch[u][id];

}

return val[u];

}

};

Trie trie;

#### 6.AC自动机

struct Trie{

int next[500010][26],fail[500010],end[500010];

int root,L;

int newnode(){

for(int i=0; i<26; i++)

next[L][i]=-1;

end[L++]=0;

return L-1;

}

void init(){

L=0;

root=newnode();

}

void insert(char buf[]){

int len=strlen(buf);

int now=root;

for(int i=0; i<len; i++){

if(next[now][buf[i]-'a']==-1)

next[now][buf[i]-'a']=newnode();

now=next[now][buf[i]-'a'];

}

end[now]++;

}

void build(){

queue<int> Q;

fail[root]=root;

for(int i=0; i<26; i++)

if(next[root][i]==-1)

next[root][i]=root;

else{

fail[next[root][i]]=root;

Q.push(next[root][i]);

}

while(!Q.empty()){

int now=Q.front();

Q.pop();

for(int i=0; i<26; i++){

if(next[now][i]==-1)

next[now][i]=next[fail[now]][i];

else{

fail[next[now][i]]=next[fail[now]][i];

Q.push(next[now][i]);

}

}

}

}

int query(char buf[]){

int len=strlen(buf);

int now=root;

int res=0;

for(int i=0; i<len; i++){

now=next[now][buf[i]-'a'];

int temp=now;

while(temp!=root){

res+=end[temp];

end[temp]=0;

temp=fail[temp];

}

}

return res;

}

void debug(){

for(int i=0; i<L; i++){

printf("id = %3d,fail = %3d,end = %3d,chi = [",i,fail[i],end[i]);

for(int j=0; j<26; j++)

printf("%2d",next[i][j]);

printf("]\n");

}

}

};

#### 7.后缀自动机

struct SuffixAutoMachine{

int ch[maxn][26];

int len[maxn],link[maxn];

int size,last;

void init(){

size=last=0;

len[0]=0; link[0]=-1;

memset(ch[0],-1,sizeof ch[0]);

}

void extend(int c){

int p=last,np=++size;

len[np]=len[p]+1;

memset(ch[np], -1, sizeof ch[np]);

while(~p&&ch[p][c]==-1)

ch[p][c]=np,p=link[p];

if(p==-1)

link[np]=0;

else{

int q=ch[p][c];

if(len[q] != len[p] + 1){

int nq = ++size;

memcpy(ch[nq], ch[q], sizeof ch[q]);

len[nq] = len[p] + 1;

link[nq] = link[q];

link[q] = link[np] = nq;

while(~p && ch[p][c] == q)

ch[p][c] = nq, p = link[p];

}

else link[np] = q;

}

last = np;

}

int find(char \*s){

bug()

int lenn=strlen(s);

int res=0,tmp=0;

int u=0;

for(int i=0; i<lenn; i++){

int c=s[i]-'a';

if(~ch[u][c])

tmp++,u=ch[u][c];

else{

while(~u&&ch[u][c]==-1)

u=link[u];

if(~u)

tmp=len[u]+1,u=ch[u][c];

else

tmp=0,u=0;

}

res=max(res,tmp);

}

return res;

}

}sam;

char s[maxn],t[maxn];

int main(){

while(scanf("%s%s",s,t)!=EOF){

sam.init();

int len=strlen(s);

for(int i=0; i<len; i++)

sam.extend(s[i]-'a');

printf("%d\n",sam.find(t));

}

return 0;

}